PROG03: Lập trình SSH LOGGER

**Lý thuyết:**

**1. Tìm hiểu về cách hoạt động của dịch vụ sshd và tiến trình ssh**

**- Cách hoạt động của dịch vụ sshd:**

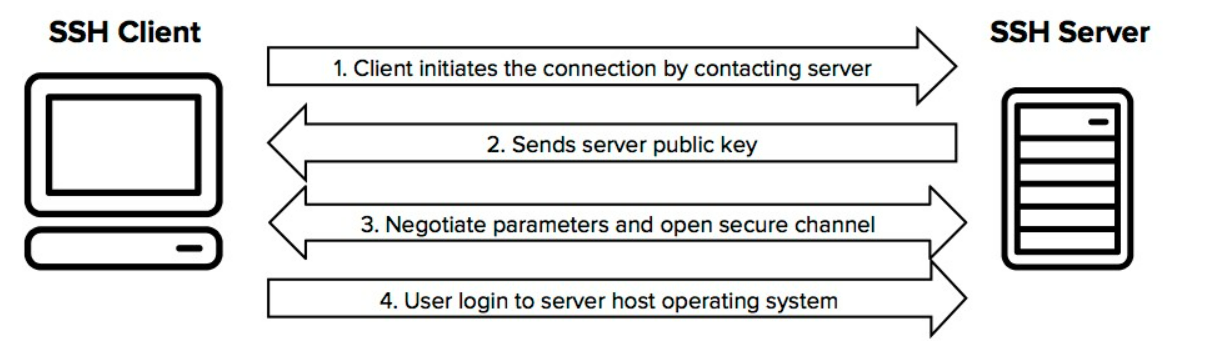
+ sshd là một tiến trình máy chủ của OpenSSH. Nó lắng nghe các kết nối đến bằng giao thức SSH và hoạt động như một máy chủ cho giao thức này. Nó xử lý xác thực người dùng, quản lý mã hóa, kết nối dầu cuối, truyền tệp và tunneling.

+ Quá trình sshd được bắt đầu khi hệ thống khởi động. Thông thường chương trình được đặt tại tệp /urs/bin/sshd. Nó chạy dưới dạng root. Quá trình ban đầu hoạt động như một máy chủ chính lắng nghe các kết nối đến. Quy trình này là quy trình có ID thấp nhất hoặc quy trình chạy lâu nhất. Nó là quy trình mẹ của tất cả các quy trình sshd khác. Các quy trình sshd khác là các quy trình con phục vụ một kết nối duy nhất. Một quy trình mới được tạo cho mỗi phiên SSH mới.

+ Máy chủ SSH có tệp cấu hình thường là /etc/sshd/sshd\_config. Tệp cấu hình chỉ định các tùy chọn mã hóa, tùy chọn xác thực, vị trí tệp, ghi nhật ký và một số thông số khác.

+ Máy chủ SSH sử dụng hệ thống con nhật ký hệ thống để ghi nhật ký. Hệ thống con nhật ký hệ thống có thể được định cấu hình để ghi dữ liệu liên quan đến SSH vào các tệp trong tệp /var/log/auth theo mặc định trên hầu hết các hệ thống.

**- Cách hoạt động của tiến trình ssh:**



+ SSH: Hay còn gọi là Secure Shell là một giao thức mạng mang tính bảo mật, cung cấp cách thức truy cập từ xa vào một thiết bị khác một cách an toàn thông qua cơ chế mã hóa. Bằng cách sử dụng một số kỹ thuật mã hóa, xác thực mỗi bên với nhau để đảm bảo dữ liệu trao đổi giữa các máy trong phiên làm việc được mã hóa và giải mã nhằm mục đích giữ vững tính toàn vẹn của dữ liệu và bảo vệ an toàn thông tin.

+ SSH hoạt động theo mô hình client-server để xác thực hai bên và mã hóa dữ liệu giữa chúng. Client sẽ là thành phần tạo phiên kết nối SSH đến thành phần khác, còn Server sẽ cho phép Client mở phiên SSH kết nối vào chính mình.

+ Server sẽ lắng nghe trên một port được chỉ định cho SSH, mặc định là port 22. Mỗi kết nối SSH bao gồm hai việc xác thực: xác thực SSH Server và xác thực SSH Client.

* Client chịu trách nhiệm khởi tạo quá trình bắt tay TCP với Server, thương lượng kết nối an toàn, xác thực Server và cung cấp thông tin đăng nhập để Server xác thực mình.
* Server có trách nhiệm thương lượng kết nối an toàn, xác thực bên Client và sinh ra phiên làm việc nếu thông tin đăng nhập được chấp nhận.

**+ Đối với SSH version 1 hoạt động có thể được chia làm 3 bước chính:**

1. Khởi tạo 1 kết nối SSH, chính là tạo ra 1 kênh giao tiếp bảo mật giữa Server và Client.
2. Client xác thực Server. Sau khi Client xác định được định danh của Server, 1 kết nối bảo mật đối xứng được hình thành giữa 2 bên.
3. Server xác thực Client dựa trên kết nối được thiết lập ở trên.

**Bước 1:** Khi Client khởi tạo 1 kết nối TCP tới 1 port ở Server, sẽ có 2 thông tin được trao đổi:

* Phiên bản Protocol mà Server hỗ trợ
* Phiên bản của gói cài đặt trên Server

Nếu phiên bản Protocol của Server phù hợp với Client thì kết nối mới được tiếp tục. Ngay sau khi Client tiếp tục kết nối, Server và Client sẽ chuyển sang sử dụng bộ giao thức Binary Packet Protocol dài 32 bit.

**Bước 2:** Server sẽ gửi các thông tin quan trọng tới Client, đây là những thông tin nắm vai trò chính trong phiên làm việc hiện tại:

* Server cho Client biết định danh của mình bằng RSA Public Key trên Server. Mọi Client kết nối tới Server đều nhận được RSA Public Key giống nhau.
* Server Key. Server Key này được tái tạo theo mỗi khoảng thời gian nhất định.
* 1 chuỗi 8 bytes ngẫu nhiên được gọi là checkbytes. Sau này Client sẽ sử dụng chuỗi này để hồi đáp lại cho Server.
* Cuối cùng, Server cấp cho Client biết toàn bộ các phương thức xác thực, mã hóa mà Server hỗ trợ.

Dựa vào danh sách các phương thức mã hóa mà Server hỗ trợ, Client tạo ra 1 Symmetric Key ngẫu nhiên và gửi cho Server. Key này sẽ được sử dụng để mã hóa và giải mã trong toàn bộ quá trình giao tiếp giữa 2 bên trong session hiện tại, đây còn được gọi là Session Key.

Để xác thực Server thì Client phải chắc chắn Server có khả năng giải mã được Session Key. Bởi vậy sau khi gửi Session Key xong Client sẽ chờ hồi âm từ phía Server. Nếu nhận được thông báo xác nhận Client đã sẵn sàng để xác thực được Server thì quá trình này hoàn tất.

**Bước 3:** Server xác thực Client: có nhiều phương pháp, trong đó có 2 phương pháp phổ biến nhất:

1. Password Authentication: Đây là phương pháp phổ biến và đơn giản nhất. Server yêu cầu Client cung cấp user name và password. Server xác nhận password dựa trên cơ sở dữ liệu được lưu trên Server.
2. Public Key Authentication: Để sử dụng Public Key Authentication, Client cần phải tạo ra RSA Public Key và RSA Private Key, 2 Key này chỉ dùng để xác thực Client. Public Key này sẽ được gửi đến Server và được Server lưu lại. Bất kỳ dữ liệu nào được mã hóa bằng Public Key chỉ có thể giải mã bằng Private Key tương ứng.

Sau khi quá trình Server chứng thực Client thành công kết nối SSH đã sẵn sàng, Client và Server cũng đã có thể gửi các gói tin cho nhau

**SSH version 2:**

- Là giao thức SSH mặc định được sử dụng ngày nay bởi một số tiến bộ trong version 2 so với version 1 như cải tiến các chuẩn mã hóa, Public Key Certification, Message Authentication Code tốt hơn và phân bố lại Session Key.

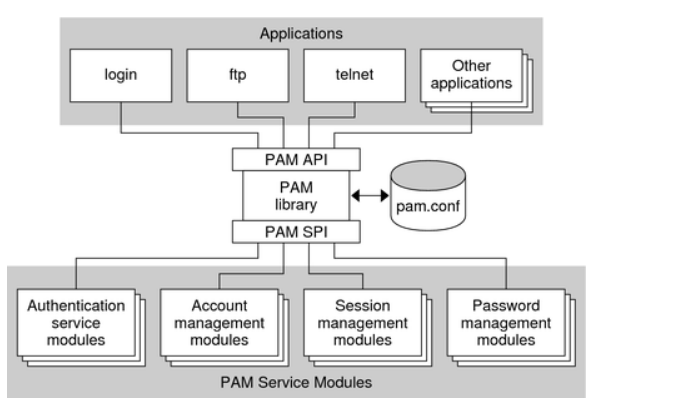
- Trong hoạt động của SSH version 1, sau khi chọn một thuật toán trong danh sách từ Server hỗ trợ, Client tạo 1 Session Key, mã hóa kép và gửi đến Server, mã hóa lần đầu bởi Public Key, mã hóa lần 2 bởi Server Key. Trong SSH version 2, không có khái niệm Server Key. Thay vào đó, Server cung cấp thêm danh sách các phương thức trao đổi khóa mà nó hỗ trợ, từ đó Client chọn một phương thức. SSH version 2 hoạt động dựa trên phương thức diffie-hellman-group1-sha1 và diffie-hellman-group14-sha1 để trao đổi khóa.

- Rekeying Session Key: Trong SSH version 1, chỉ có một Session Key được sử dụng cho toàn bộ session làm việc của SSH. Trong SSH version 2 có bổ sung tính năng Rekeying Session Key cho phép thay đổi Session Key mà không làm mất session làm việc hiện tại.

**2. Tìm hiểu cơ chế xác thực pam của hệ điều hành Linux**

- **PAM (Pluggable Authentication Modules)** là một bộ các thư viện chia sẻ cho phép người quản trị hệ thống cục bộ chọn cách các ứng dụng xác thực người dùng. Chức năng của (các) tệp cấu hình là cung cấp ánh xạ từ tên dịch vụ của ứng dụng tới một tập hợp câc module cung cấp dịch vụ xác thực cho ứng dụng thô. Khi một ứng dụng nhận biết pam với một tập tin trong /etc/pam.d bắt đầu, thư viện PAM nạp cấu hình cho dịch vụ được chỉ định và xây dựng bốn chuỗi module (một cho mỗi cơ sở). Thông qua việc sử dụng các module, PAM có thể cho phép một chương trình tìm kiếm thông tin qua một số cơ sở dữ liệu mật khẩu khác nhau, ngay cả khi chương trình đó không được mã hóa rõ ràng cho cơ sở dữ liệu cụ thể đó

**- Cấu trúc của PAM:**

****

+ Thư mục /etc/pam.d/ chứa các tệp cấu hình PAM cho mỗi ứng dụng PAM-Aware (login, passwd, sshd, vsftp,...)

Mỗi ứng dụng hoặc dịch vụ PAM-aware có một tệp trong thư mục /etc/pam.d/. Mỗi tệp trong thư mục này có cùng tên với dịch vụ mà nó kiểm soát truy cập. Chương trình PAM-Aware chịu trách nhiệm xác định tên dịch vụ của mình và cài đặt tệp cấu hình PAM của riêng mình trong thư mục /etc/pam.d/. Ví dụ: chương trình login xác định tên dịch vụ của nó là login và cài đặt tệp cấu hình /etc/pam.d/login PAM.

+ Thư viện các module được lưu trong /lib/security (pam\_chroot.so, pam\_access.so, pam\_rootok.so, pam\_deny.so,...)

+ Các file cấu hình được lưu trong /etc/security (access.conf, chroot.conf, group.conf,...)

* **access.conf:** Điều khiển quyền truy cập, được sử dụng cho thư viện pam\_access.so
* **group.conf:** Điều khiển nhón người dùng, sử dụng bởi pam\_group.so
* **limits.conf:** Thiết lập các giới hạn tài nguyên hệ thống, được sử dụng bởi pam\_limits.so
* **pam\_env.conf:** Điều khiển khả năng thay đổi các biến môi trường, sử dụng cho thư viện pam\_env.so
* **time.conf:** thiết lập hạn chế thời gian cho dịch vụ và quyền người dùng, sử dụng cho thư viện pam\_time.so

**- Phương thức hoạt động của PAM:**

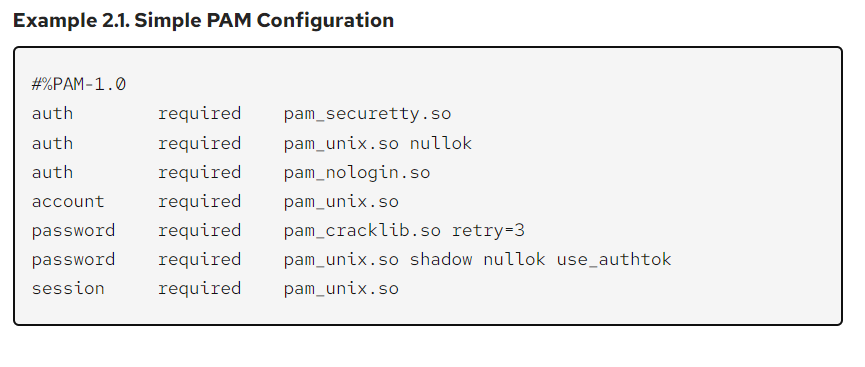
Các chương trình login, pass, su, sudo,... trên được gọi là privilege-granting application (chương trình trao đặc quyền). PAM-aware application: là chương trình giúp các privilege-granting application làm việc với thư viện PAM.

**Các bước hoạt động:**

* **Bước 1:** Người dùng chạy một ứng dụng để truy cập vào dịch vụ mong muốn. Ví dụ: login
* **Bước 2:** PAM-aware application gọi thư viện PAM để thực hiện nhiệm vụ xác thực
* **Bước 3:** PAM library sẽ dựa vào file cấu hình của chương trình đó trong /etc/pam.d (ví dụ login thì file cấu hình là /etc/pam.d/login) xác định loại xác thực nào được yêu cầu cho chương trình trên. Trong trường hợp không có file cấu hình, thì file /etc/pam.d/other sẽ được sử dụng.
* **Bước 4:** PAM library sẽ load các module yêu cầu cho xác thực trên.
* **Bước 5:** Các modules này sẽ tạo một liên kết tới các hàm chuyển đổi (conversation functions) trên chương trình.
* **Bước 6:** Các hàm này dựa vào các modules mà đưa ra các yêu cầu với người dùng. Ví dụ yêu cầu người dùng nhập password.
* **Bước 7:** Người dùng nhập thông tin vào theo yêu cầu
* **Bước 8:** Sau khi quá trình xác thực kết thúc, chương trình này sẽ dựa vào kết quả mà đáp ứng yêu cầu người dùng (ví dụ cho phép login vào hệ thống) hay thông báo thắt bại với người dùng.

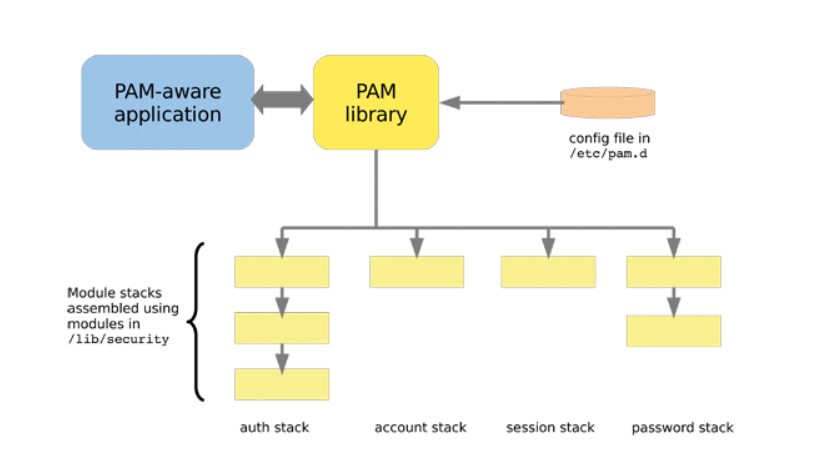
**- Định dạng tệp cấu hình PAM:**

**Ví du:**

****

*Module\_interface* *control\_flag* *module\_name* *module\_arguments*

**PAM Module Interfaces**

****

Bốn loại PAM Module Interfaces có sẵn:

* **auth:** thực hiện xác thực. Thông thường, một auth module sẽ yêu cầu password để kiểm tra, hay thiết lập các định danh như nhóm người dùng hay thẻ kerberos.
* **account:** xác minh rằng quyền truy cập được phép. Ví dụ: nó kiểm tra xem tài khoản người dùng đã hết hạn chưa hoặc người dùng có được phép đăng nhập vào một thời điểm cụ thể trong ngày hay không.
* **password:** sử dụng để thay đổi mật khẩu người dùng
* **session:** cấu hình và quản lý các phiên người dùng. Các module với giao diện này cũng có thể thực hiện các tác vụ bổ sung cần thiết để cho phép truy cập, chẳng hạn như gắn thư mục chính của người dùng và làm cho hộp thư của người dùng khả dụng.

**PAM Control Flags**

Tất cả các modules PAM tạo ra kết quả thành công hoặc thất bại. Control Flags nói với PAM nên làm gì với kết quả

* **required:** cờ diều khiển này nói với PAM library yêu cầu sự thành công của modules tương ứng. Nếu không thành công tại thời điểm này, người dùng không được thông báo cho đến khi kết quả của tất cả các thử nghiệm module tham chiếu rằng giao diện đã hoàn tất.
* **sufficient:** Kết quả module bị bỏ qua nếu nó thất bại. Nếu kết quả module được gắn cờ sufficient là thành công và không có modules nào trước đó được gắn cờ required mà thất bại thì không có kết quả nào khác được yêu cầu và người dùng được xác thực với dịch vụ.
* **requisite:** cờ này nói với PAM library loại bỏ ngay quá trình xác thực khi gặp bất kỳ thông báo thất bại của module nào.
* **optional:** Kết quả modlue bị bỏ qua. Một module được gắn cờ là tùy chọn chỉ trở nên cần thiết để xác thực thành công khi không có module nào khác tham chiếu giao diện.

**PAM Module Names:**

Cho PAM biết module nào sẽ được sử dụng. Tên thư mục bị bỏ qua vì ứng dụng được liên kết với phiên bản thích hợp của libpam, có thể định vị đúng phiên bản của module.

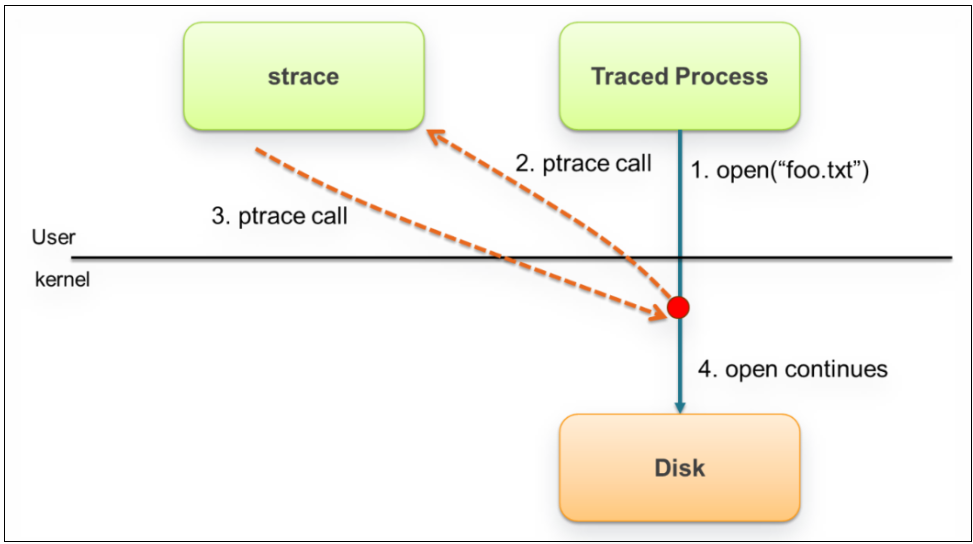
**PAM Module Arguments:**

PAM sử dụng các đối số để truyền thông tin đến module có thể cắm trong quá trình xác thực cho một số module.

**Danh sách các tùy chọn module có khả năng được nhận dạng bởi tất cả các module:**

* **debug:** Sử dụng lệnh syslog để ghi thông tin gỡ lỗi vào các tệp nhật ký hệ thống
* **use\_first\_pass:** Module không nên nhắc người dùng nhập mật khẩu. Nó sẽ lấy mật khẩu đã nhập trước đó (từ module auth trước đó) và sử dụng nó. Nếu điều đó không hiệu quả thì người dùng sẽ không được xác thực (tùy chọn này chỉ dành cho module xác thực mật khẩu).
* **try\_first\_pass:** Module sẽ thử xác thực bằng mật khẩu đã nhập trước đó (từ module auth trước đó). Nếu điều đó không có tác dụng thì người dùng sẽ được nhắc nhập mật khẩu (tùy chọn này chỉ dành cho các module auth).

**3. Tìm hiểu cách hoạt động của strace, ptrace**

****

**-** Mọi tiến trình chạy trên máy tính để bàn đều sử dụng lệnh gọi hệ thống để giao tiếp với hệ điều hành. Sử dụng strace,có thể theo dõi các cuộc gọi hệ thống như vậy một cách dễ dàng.

- Khi các chương trình chạy trên Linux muốn sử dụng các tài nguyên do hệ điều hành quản lý (đọc tệp, tạo quy trình, v.v.), chúng sẽ thực hiện lệnh gọi hệ thống đến hệ điều hành. Các lệnh gọi hệ thống hoạt động ở cấp độ kenel và thực hiện các hoạt động cần thiết, để lại quyền kiểm soát cho chương trình gọi. Công cụ strace cung cấp khả năng theo dõi các lệnh gọi hệ thống này trên Linux.

- strace là một tiện ích không gian người dùng chẩn đoán, gỡ lỗi và hướng dẫn cho Linux. Nó được sử dụng để giám sát và xáo trộn các tương tác giữa các quy trình và nhân Linux, bao gồm các lệnh gọi hệ thống, phân phối tín hiệu và các thay đổi của trạng thái quy trình. **Hoạt động của strace có thể thực hiện được nhờ tính năng hạt nhân được gọi là ptrace.**

- ptrace có thể làm nhiều việc phức tạp và được sử dụng, chẳng hạn như bởi các trình gỡ rối như gdb để xem xét một quy trình đang chạy. Strace sử dụng nó để thiết lập một quy trình mục tiêu và “lắng nghe” các lệnh gọi hệ thống của quy trình đó.

- strace hoạt động bằng cách sử dụng lệnh gọi hệ thống ptrace khiến hạt nhân tạm dừng chương trình đang được theo dõi mỗi khi nó vào hoặc ra khỏi hạt nhân thông qua lệnh gọi hệ thống. Sau đó, chương trình truy tìm (trong trường hợp này là strace) có thể kiểm tra trạng thái của chương trình bằng cách sử dụng ptrace.

- strace nhận các đối số cho mỗi lệnh gọi hệ thống tùy thuộc vào cách hệ thống đó hoạt động. Trên các hệ thống x86-64, các đối số cho các lệnh gọi hệ thống được chuyển vào các thanh ghi CPU. Trong trường hợp này, strace có thể gọi ptrace với đối số PTRACE\_GETREGS để nhận bản sao của các giá trị thanh ghi và in chúng.

**Thực hành:**

Trường hợp đang có quyền root để truy cập vào một máy tính linux. Xây dựng chương trình để lấy được username, password ssh trong 2 trường hợp:

**- Trường hợp 1:** Một user tiến hành truy cập vào máy tính đó thông qua ssh. Chương trình **sshtrojan1.sh.**

+ Mô tả lại cách hoạt động, cách sử dụng của sshtrojan1:

1. Kiểm tra xem có đang chạy chương trình bằng quyền root hay không.
2. Kiểm tra file log đã tồn tại hay chưa. Nếu chưa tồn tại thì tạo mới file.
3. Thực thi file script để hiển thị được username và password ssh
4. Chạy tệp cấu hình PAM của sshd. PAM ghi mật khẩu vào stdin của tập lệnh và cung cấp tên người dùng làm biến môi trường. Sau đó đầu ra của path\_script được thêm vào file\_log.
5. Khởi động lại dịch vụ SSH.

**- Trường hợp 2:** Một người dùng đứng trên máy tính đó ssh đi một máy tính khác. Chương trình **sshtrojan2.sh**

**+** Mô tả lại cách hoạt động, cách sử dụng của sshtrojan2:

1. Kiểm tra xem có đang chạy chương trình bằng quyền root hay không.
2. Kiểm tra file log đã tồn tại hay chưa. Nếu chưa tồn tại thì tạo mới file.
3. Phân tách lấy PID
4. Lấy username từ tiến trình ssh
5. Lấy nhật ký theo dõi hệ thống của PID
6. Phân tách lấy mật khẩu từ systrace log
7. Lưu username và password vào file log.